

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 0

## 2

3

## 4

5

## 6

7

1 when the number of entries exceeds a pre-allocated storage limit.  
2 This causes entries to be overwritten and thus lost before  
3 training can occur. Furthermore, if the audit logs are stored  
4 within the database itself (to be used later during an  
5 operational step), such storage uses up valuable resources on the  
6 database and negatively impacts the database's performance.  
7

8 Examples of offline non-real-time database intrusion  
9 detection systems are described in *Lee, et al.*, "Learning  
10 Fingerprints for a Database Intrusion Detection System", ESORICS  
11 2002, pp. 264-279, published in November 2002 by Springer-Verlag,  
12 Berlin and Heidelberg, Germany; and *C. Chung, et al.*, "DEMIDS: A  
13 Misuse Detection System for Database Systems", Department of  
14 Computer Science, University of California at Davis, Davis,  
15 California, October 1, 1999.  
16

#### 17 Disclosure of Invention

18 The present invention comprises systems, methods, and  
19 computer-readable media for training a computer code intrusion  
20 detection system in real time. A method embodiment of the  
21 present invention comprises the steps of observing (22), in real  
22 time, commands (5) that are accessing the computer code (1); and  
23 deriving (23) from said commands (5), in real time, a set (6) of  
24 acceptable commands.  
25  
26  
27  
28

### Brief Description of the Drawings

These and other more detailed and specific objects and features of the present invention are more fully disclosed in the following specification, reference being had to the accompanying drawings, in which:

Figure 1 is a block diagram illustrating embodiments of the present invention.

Figure 2 is a flow diagram illustrating method embodiments of the present invention.

### Detailed Description of the Preferred Embodiments

This invention has applicability to any code intrusion detection system, i.e., any system in which computer code 1 is susceptible to being attacked by commands 5 which may be malicious, due to malicious intent on the part of the user 3 who issues the command 5. As used herein, "user" can refer to a client computer 3 and/or to a human who has control of computer 3. As illustrated in Figure 1, there can be a plurality N of users 3, where N is any positive integer.

Most of the following description illustrates the special case where the computer code 1 is a database 1. Database 1 can be any type of database, such as a relational database or a flat file. When database 1 is a relational database, commands 5 are typically written in a SQL language. As used herein, "SQL" is taken in the broad sense to mean the original language known as

1 SQL (Structured Query Language), any derivative thereof, or any  
2 structured query language used for accessing a relational  
3 database. In the case where computer code 1 is not a relational  
4 database, the commands can be written in another language such as  
5 XML. Database 1 may have associated therewith an internal audit  
6 table 11 and/or an external database log file 12 for storing  
7 audit and/or ancillary information pertaining to database 1.  
8 Database 1 is typically packaged within a dedicated computer  
9 known as a database server 2, which may also contain database  
10 communications module 15 and other modules not illustrated.  
11 Modules 1, 11, 12, 15, 13, 17, 4, 7, 6, 8, and 9 can be  
12 implemented in software, firmware, hardware, or any combination  
13 thereof, and are typically implemented in software.  
14  
15

16 Computer code intrusion detection system 19 (and its special  
17 case, database intrusion detection system 19) encompasses modules  
18 17, 4, 13, 7, 6, 8, and 9. Figure 1 illustrates the case where  
19 these modules are stand-alone modules separate from database  
20 server 2. However, they could just as well be incorporated  
21 within database server 2, e.g., they could be incorporated within  
22 database communications module 15. Thus, intrusion detection  
23 system 19 could be published by a third party as a standalone  
24 package on any type of computer-readable medium, or bundled by  
25 the manufacturer of the database 1 with module 15. The purpose  
26 of intrusion detection system 19 is to protect computer code 1  
27  
28

1 from users 3 that have nefarious intent. For example, such users  
2 may desire to steal credit card information from database 1,  
3 modify database 1 to credit their account with additional money,  
4 modify database 1 to spread a computer worm, etc.

5  
6 With respect to Figure 2, it can be seen that intrusion  
7 detection system 19 operates in three phases. Phase one is a  
8 training phase and encompasses steps 21 through 24. Phase two is  
9 an optional reporting phase and encompasses step 25. Phase 3 is  
10 an operational phase and encompasses steps 26 through 29.

11 In order for intrusion detection system 19 to be used, it  
12 must be first trained. The training phase is initiated at step  
13 21. This is done by system administrator 10 flipping a switch  
14 (which may be located, for example, on database server 2 or on  
15 training module 4); by means of a preselected event occurring  
16 (e.g., the first of each month or the addition of a new table  
17 within database 1); or by any other means known to one of  
18 ordinary skill in the art for starting a software or other  
19 computer system.  
20

21 At step 22, training module 4 observes, in real time,  
22 commands 5 that users 3 send to database 1. As used herein,  
23 "real time" means "during a short time interval surrounding an  
24 event". Thus, observing a command 5 in real time means that the  
25 command 5 is observed during a short time interval surrounding  
26 the instant that the command 5 enters the database 1.  
27  
28

1 Examples of commands 5 include querying the database 1,  
2 adding an entry to the database 1, deleting an entry from the  
3 database 1, and modifying an entry in the database 1. There are  
4 two major ways in which the observing step 22 is performed: real-  
5 time auditing and in-line interception.  
6

7 Real-time auditing is typically used in cases where database  
8 1 has an auditing feature. The auditing information may be  
9 placed into an audit table 11 internal to database 1 or into an  
10 external database log file 12. In real-time auditing, training  
11 module 4 instructs the database 1 to generate a new event every  
12 time a command enters database 1. Each event can include such  
13 items as the text of the command 5, a date/time stamp,  
14 information on the user that issues the command 5, the IP  
15 (Internet Protocol) address of the issuing computer 3, the  
16 application that issued the command 5, etc.  
17

18 The event data can appear in string or binary form, and can  
19 be extracted using a number of different techniques, depending on  
20 the implementation of the IDS 19, including APIs (Application  
21 Programming Interfaces) that access the computer code 1. One  
22 example is to use ODBC (Open DataBase Connectivity), a set of C  
23 language API's that allows one to examine or modify data within  
24 database 1. If the Java programming language is used, JDBC (Java  
25 DataBase Connectivity) can be used instead. Another way of  
26 extracting the needed information from database 1 is to use code  
27  
28

1 injection or patching to inject logic into one or more modules  
2 1,15 within database server 2, to transfer control to training  
3 module 4. In another embodiment, called "direct database  
4 integration", the database 1 vendor, who has access to the  
5 commands 5 in conjunction with the normal operation of the  
6 database 5, makes the commands 5 available to intrusion detection  
7 system 19. In yet another embodiment, in cases where database 1  
8 supports it, external database log file 12 may be examined  
9 without the need to resort to special software. Once an event  
10 has been processed by the IDS 19, it can optionally be expunged  
11 from any table or log file it is stored in, to make room for  
12 subsequent events.  
13

14 In in-line interception, at least one of a proxy, firewall,  
15 or sniffer 13 is interposed between database 1 and users 3. The  
16 proxy, firewall, or sniffer 13 examines packets of information  
17 emanating from users 3 and extracts the relevant information  
18 therefrom. Proxy, firewall, or sniffer 13 may need to decrypt  
19 the communications emanating from users 3 if these communications  
20 are encrypted.  
21

22 After a command 5 has been captured in step 22, at step 23  
23 training module 4 analyzes the command 5 and updates a set 6 of  
24 acceptable commands, again in real time. In other words, set 6  
25 is updated command-by-command. In one embodiment, the deriving  
26 step 23 comprises grouping the commands 5 into categories and  
27  
28

1 updating statistical information pertaining to the categories,  
2 all in real time. The categories are pre-selected, and can be  
3 fed to training module 4 via prestored but modifiable training  
4 parameters 17. The categories can include one or more of the  
5 following: canonicalized commands; the dates and times at which  
6 the commands access the computer code 1; logins (user IDs,  
7 passwords, catch phrases, etc.) of users 3 issuing the commands  
8 5; the identities of users 3 issuing the commands 5; the  
9 departments of the enterprise in which the users 3 work, or other  
10 groups to which the users 3 belong; the applications (i.e.,  
11 software programs or types of software programs) that issue the  
12 commands 5; the IP addresses of the issuing computers 3;  
13 frequency of issuing a given command 5 by a given user 3;  
14 identities of users 3 accessing a given field or fields within  
15 the computer code 1; the times of day that a given user or group  
16 of users 3 accesses a given field or fields within the computer  
17 code 1; the fields or combination of fields being accessed by  
18 given commands 5; tables or combinations of tables within the  
19 computer code 1 accessed by the commands.

22       The invention will now be further described with respect to  
23 the special case in which the commands 5 are grouped into  
24 categories comprising canonicalized commands. A canonicalized  
25 command is a command 5 stripped of its literal field data. Thus,  
26 for example, let us assume that the command 5 is:  
27  
28

1 SELECT NAME FROM PATIENTS WHERE NAME LIKE 'FRANK' AND AGE > 25

2       Literal field data is defined as a specific value of a  
3 parameter. In this case, the literal field data is "FRANK" and  
4 "25". Thus, a canonicalized form of the command 5 is:

5       SELECT NAME FROM PATIENTS WHERE NAME LIKE \* AND AGE > \*

6       Literal fields can include literal numbers (plain numbers),  
7 dates, times, strings, and potentially named ordinal values  
8 (symbolic words used to represent numbers, e.g., "January"  
9 represents the first month, "Finance" represents department 54,  
10 etc.). Training module 4 uses the canonicalized command as an  
11 index to group and categorize the commands 5. Even in a large  
12 database 1, the number of command categories is usually less than  
13 20,000 or so.

14  
15       To continue with our example, the following commands 5 would  
16 all be placed into the same command category:

17  
18 SELECT NAME FROM PATIENTS WHERE NAME LIKE 'FRANK' AND AGE > 25

19 SELECT NAME FROM PATIENTS WHERE NAME LIKE 'BILL' AND AGE > 40

20 SELECT NAME FROM PATIENTS WHERE NAME LIKE 'ED' AND AGE > 25

21       In one embodiment, the command

22 SELECT NAME FROM PATIENTS WHERE NAME IS 'FRANK' AND AGE > 25

23 would be treated differently from the command

24 SELECT NAME FROM PATIENTS WHERE NAME LIKE 'FRANK' AND AGE > 25.  
25  
26  
27  
28

1 In an alternative embodiment, a list of synonyms is kept, so that  
2 "IS" is treated the same as "LIKE", and thus these two commands  
3 are treated as falling within the same command category.

4 In the last example, instead of using "ED", one could use  
5 "ED%" where % is a wildcard character. (Other wildcard  
6 characters might include #, ?, or \_, depending upon the language  
7 variant used.) Thus, "ED%" would pick up "EDWARD", "EDDIE", and  
8 "EDWINA", as well as "ED". Wildcards are tracked by field and by  
9 command category, to give information needed to determine whether  
10 the use of the wildcard is innocuous or suspicious (see the  
11 following discussion with respect to step 25).

12 The fact that a wildcard is not observed in a given field  
13 during the training phase can be saved in the set 6 of acceptable  
14 commands, along with canonicalized data as described above.  
15 Then, during the operational phase, the presence of a wildcard  
16 character in that field is flagged as being suspicious.

17 For a given command 5, training module 4 examines the value  
18 of each literal field and determines whether its value is in the  
19 set of values seen thus far for that field, or whether it  
20 constitutes a new value. Continuing with our example based upon  
21 the three commands 5 given above, training module 4 stores the  
22 following information in NAME and AGE fields associated with the  
23 given canonicalized command category:

24 CANONICALIZED COMMAND CATEGORY:

25 SELECT NAME FROM PATIENTS WHERE NAME LIKE \* AND AGE > \*

1 NAME statistics:

2 Total number of different field values: 3 (Frank, Bill,  
3 and Ed)

4 Generalize flag: FALSE (Do not generalize)

5 Wildcard flag: TRUE (wildcards have been used in this  
6 field, e.g. "ED")

7 AGE statistics:

8 Total number of different field values: 2 (25, 40)

9 Generalize flag: FALSE (Do not generalize)

10 Wildcard flag: FALSE (wildcards have not been used in  
11 this field)

12 Notice from the above that the actual names are stored  
13 because there aren't too many. Alternatively, all names having  
14 fewer than a preselected number of characters could be stored.

15 In one embodiment, whenever, for a given literal field, the  
16 number of observed values is greater than or equal to a  
17 preselected threshold value S, the training module 4 sets a flag  
18 for this field in this canonicalized command category, indicating  
19 that the field should be generalized, allowing any legitimate  
20 value. Thus, for example, if S=4 and training module 4 observes  
21 a fourth query:

22 SELECT NAME FROM PATIENTS WHERE NAME LIKE 'STEVE' AND AGE > 25,  
23 training module 4 determines that it has already seen three  
24 distinct values (Frank, Bill, and Ed) in the NAME field, and it  
25 has just been presented with Steve. This now pushes the number  
26 of distinct cases for the NAME field to four, which meets or  
27 exceeds the preselected threshold. In this case, training module  
28 4 automatically sets the Generalize flag for this field in this

1 command category, indicating that any value is acceptable. The  
2 entry then changes to:

3 CANONICALIZED COMMAND CATEGORY:

4 SELECT NAME FROM PATIENTS WHERE NAME LIKE \* AND AGE > \*

5 NAME statistics:

6 Total number of different field values: 4 or more (no  
actual field data saved)

7 Generalize flag: TRUE (Allow any valid string for this  
field)

8 Wildcard flag: TRUE (wildcards have been used in this  
field, e.g. "ED")

9 AGE statistics:

10 Total number of different field values: 2 (25, 40)

11 Generalize flag: FALSE (Do not generalize)

12 Wildcard flag: FALSE (wildcards have not been used in  
this field)

13 This technique may be advantageously used when there are a  
14 constrained number of values for that field. For example, if the  
15 field represents the month of the year, there are a maximum  
16 number of 31 days in any month, so S should be set to 31.  
17 Similarly, if the field represents the year, there are a maximum  
18 of 12 months in any year, so S should be set equal to 12. Once a  
19 given field within a given canonicalized command category has  
20 been generalized, there is no need to save all observed valued  
21 for this field anymore. Thus, the code intrusion detection  
22 system 19 uses less memory as more items are generalized.

23 In addition to the field attributes "generalize" and  
24 "wildcard", other attributes e.g., "length of string", can be  
25 used. The "generalize" attribute can be a partial or conditional  
26 "generalize", e.g., any name is allowed in the NAME field as long  
27 as it contains fewer than a preselected number of characters.  
28

1       The training phase is ended, at step 24, by any one of a  
2 number of means. For example, system administrator 10 can flip a  
3 switch on database server 2 or training module 4. Alternatively,  
4 the training phase may end by a statistical technique, e.g.,  
5 training module 4 monitors the occurrence or frequency of new  
6 canonicalized command categories being established or new fields  
7 being established, and determines that a preselected frequency  
8 threshold has been met. As with all of the preselected  
9 parameters, the preselected frequency threshold may be stored in  
10 training parameters 17. Alternatively, the training phase may  
11 end by the occurrence of a preselected elapsed or absolute time,  
12 or by any other means known to one of ordinary skill in the art.  
13

14       At the conclusion of the training phase, the set 6 of  
15 acceptable commands that has been gradually built up in real time  
16 during the training phase becomes used as the basis for  
17 comparison during the subsequent operational phase (phase 3 in  
18 Figure 2). Based upon the above example, the set 6 of acceptable  
19 commands contains the following two commands:  
20

21       SELECT NAME FROM PATIENTS WHERE NAME LIKE \* AND AGE > 25

22       SELECT NAME FROM PATIENTS WHERE NAME LIKE \* AND AGE > 40

23       In the above, \* means that the field can hold any valid  
24 string value. These two commands in set 6 represent all valid  
25 SQL commands 5 observed during the training phase. This real-  
26 time system 4 has now built up a table of canonicalized commands,  
27  
28

1 determined which fields in these commands should be generalized,  
2 and has done this using a limited amount of memory. Even with  
3 20,000 commands 5, only megabytes of storage are needed. This is  
4 in contrast to the off-line non-real-time systems of the prior  
5 art where logs 11, 12 may grow to gigabytes in length. For  
6 example, in the present invention, if the following command 5 was  
7 issued 5 million times:  
8

9 SELECT NAME FROM PATIENTS WHERE NAME LIKE somename AND AGE > 25  
10 it would require less than a kilobyte to represent this command  
11 data. On the other hand, off-line non-real-time training systems  
12 of the prior art would have to keep a full audit log 11, 12 of  
13 all 5 million occurrences as input to the off-line training  
14 process.  
15

16 Returning to Figure 2, optional step 25 is invoked. When  
17 step 25 is used, suspicious activity is tracked during the  
18 training phase. For example, the number or percentage of times a  
19 wildcard character is used for a particular field for a  
20 particular command category can be tracked. If this number or  
21 percentage exceeds a preselected threshold (again, provided by  
22 parameters 17), the activity is deemed to be suspicious. This  
23 suspicious activity can then be reported to the system  
24 administrator 10. A wildcard may be either innocuous or  
25 suspicious, based upon where it is used. For example, if a  
26 wildcard is used in a NAME OF AUTHOR field in a database 1 of  
27  
28

1 library books, that is probably innocuous. On the other hand, if  
2 a wildcard is used in the PASSWORD field in a database 1 of  
3 credit card information, that most likely would be deemed  
4 suspicious. The system administrator can then selectively remove  
5 from the set 6 of acceptable commands such suspicious commands 5.

7 After the training phase and the optional suspicious  
8 activity reporting phase have been completed, the intrusion  
9 detection system 19 operates in the operational phase. At step  
10 26, commands 5 that are currently accessing database 1 are  
11 compared by comparing module 7 against commands in the set 6 of  
12 acceptable commands. Module 7 can extract the current commands 5  
13 in the same manner as described above for training module 4. The  
14 current command 5 is pre-processed similarly to the way the  
15 corresponding command 5 was pre-processed in the training phase,  
16 if this is needed to facilitate the comparison. For example, in  
17 the illustration described above, the command 5 is first  
18 canonicalized.

20 The basic rule by which module 7 operates is that if the  
21 current command 5 matches a command in the set 6 of acceptable  
22 commands, comparing module 7 allows the current command 5 to  
23 access database 1 (at step 27). If, on the other hand, the  
24 current command 5 does not match a command 5 in the set 6 of  
25 acceptable commands, comparing module 7 flags the current command  
26 5 as being suspicious (at step 28). Then a post-flagging  
27  
28

1 protocol is performed at step 29. This protocol entails  
2 execution of at least one of the following steps: an alert is  
3 sent to the system administrator 10; the command 5 is not allowed  
4 to access the computer code 1; the command 5 is allowed to access  
5 the computer code 1, but the access is limited in some way (for  
6 example, the amount of data sent back to user 3 is limited, or  
7 wildcard characters are removed from certain fields); the command  
8 5 is augmented, e.g., investigational code is inserted into the  
9 command 5 to provoke an audit trail; the user 3 sending the  
10 command 5 is investigated. Such an investigation can be  
11 performed by computer means (e.g., sending out a digital trace to  
12 determine the identity of the user 3) and/or by off-line means  
13 (sending a human private investigator to spy on user 3).  
14  
15

16 It will be seen from the above discussion that the present  
17 invention may offer, inter alia, the following advantages over  
18 prior art systems:

- 19 • It may reduce by several orders of magnitude the amount  
20 of data required to perform training of an intrusion  
21 detection system.
- 22 • It eliminates the need for the system administrator 10  
23 to collect, manage, and delete large log files.
- 24 • It eliminates problems associated with overwritten log  
25 entries, ensuring that the training never misses any  
26 legitimate training data.  
27  
28

- It improves the performance of the database 1 by eliminating the storage of audit log data within built-in audit table 11.
- It eliminates the need for the system administrator 10 to manually configure database 1 to perform audit logging.
- It is an "on line" system, i.e., it doesn't disrupt the normal operations of the database 1.

The above description is included to illustrate the operation of the preferred embodiments and is not meant to limit the scope of the invention. The scope of the invention is to be limited only by the following claims. From the above discussion, many variations will be apparent to one skilled in the art that would yet be encompassed by the spirit and scope of the present invention.

What is claimed is: